Dynamic Modelling
with Sequence Diagrams

Prof. T.H. Tse
Department of Computer Science
Email: thtse@cs.hku.hk
Web: hku.hk/thtse

Importance of Dynamic Modelling

Importance of Dynamic Modelling

Dynamic Models

- Specify ***behaviour*** of objects by describing ***sequentiality*** and ***interaction control*** of their operations
- ***Disregard***
  - what the operations do
  - how they are implemented .

Dynamic Models

- Represented graphically by
  - sequence diagrams
  - state machines

Can also be regarded as a use case model

but never as a program

Can also be regarded as a design model .

Events

- An event occurs when a message is passed
  - between an object and an external party
  - or between two objects
- Happens at a point in time and has ***no duration***
- Information may exchange through parameters .

Caller hands up

Disconnect phones

Examples of Events

Possibly with Objects and Attributes

- caller hangs up
- car has just exceeded speed limit
- caller dials (digit)
- debit (amount)
- user enters (text)
- transfer to (***Account***, amount)
- depart (***Airport***, date, time)

Current object is an implicit parameter

Current object is an implicit parameter

Current object is an implicit parameter .

Example

Scenario for Phone Call

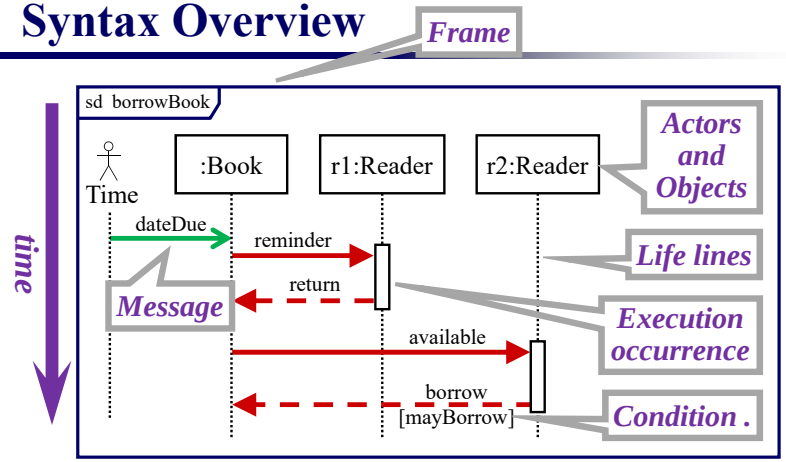
- caller lifts receiver
- dial tone begins
- caller dials digit (2)
- dial tone ends
- caller dials digit (8)
- caller dials digit (5)
- caller dials digit (9)
- caller dials digit (2)
- caller dials digit (1)
- caller dials digit (8)
- caller dials digit (0) …
- called phone begins ringing
- ringing tone appears in calling phone
- called party answers
- called phone stops ringing
- ringing tone disappears in calling phone
- connect phones
- called party hangs up
- disconnect phones
- caller hangs up .

Sequence Diagrams

- A sequence diagram is a ***graphical*** representation of related scenarios
- Shows
  - ***objects*** as vertical dotted lines
  - ***events*** as horizontal arrows from sender object to receiver object .

Sequence Diagrams

Syntax Overview



Message Types

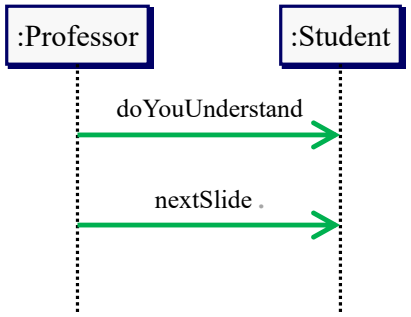
- ***Asynchronous*** message
- ***Synchronous*** message (with operation call)
- Reply message
- Object creation message
- Lost message
- Found message .

Asynchronous Messages

- Caller does ***not*** wait for the message to be handled before continuing with other messages
- ***Most*** messages in sequence diagrams are ***asynchronous*** .

Asynchronous Messages

Example 1



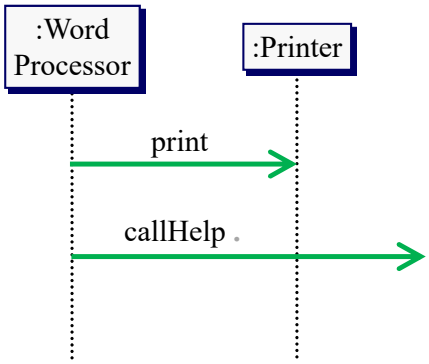
Asynchronous Messages

Example 2

- Word Processor object sends an ***asynchronous*** message to Printer object asking it to print a file
- Printer object can print the file at any time when it is free
- Word Processor object can do anything after it has sent off the ***asynchronous*** message

Asynchronous Messages

Example 2



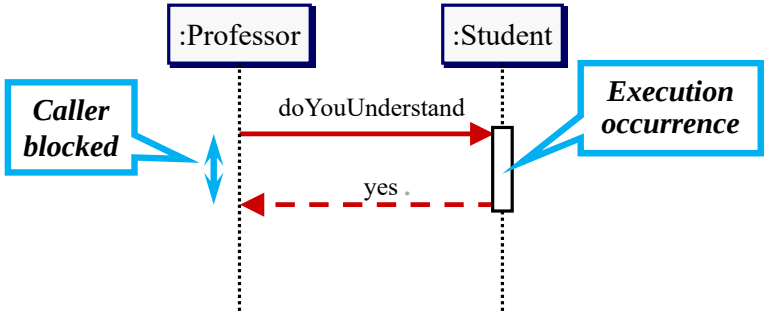
Synchronous Messages

- Caller waits for the message to be handled before continuing with other messages
- Typically implemented together with operation call

Synchronous Message and Synchronous Call

Example 1

- Model synchronous call by ***execution occurrence***



Return Values

- Indicated by dashed arrow, together with the actual value returned
- Can skip the return value when it is obvious
  - Such as **total** after calling **getTotal**

UML recommendation: Skip the obvious

My recommendation: Don't skip .

Return Values

- Must specify the return value when we need to use it elsewhere
  - Say, when **total** is used as parameter in another message

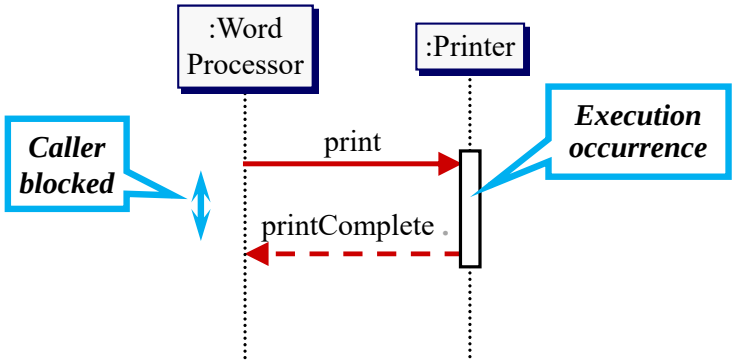
My recommendation: A lot of manual checks if you skipped the obvious .

Synchronous Message and Synchronous Call

Example 2

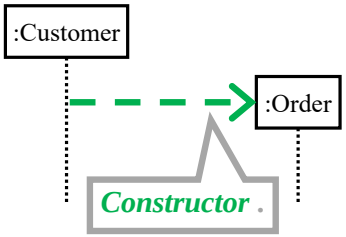
- Word Processor object sends a ***synchronous*** message to Printer object asking it to print a file
- Word Processor object does not do anything until it receives a (software) acknowledgement from Printer object saying that printing is complete .

Synchronous Message and Synchronous Call
Example 2

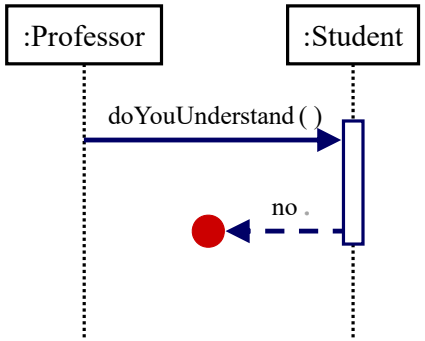


Object Creation Messages

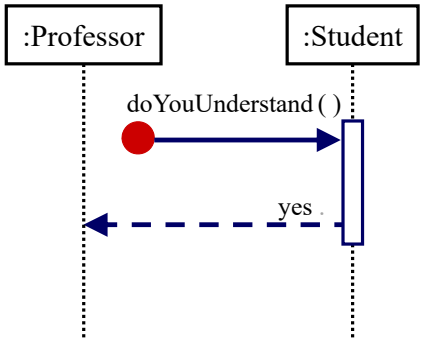
- An object may create another object via an object creation message



Lost Messages

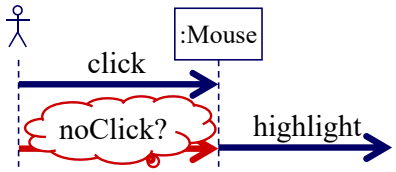


Found Messages

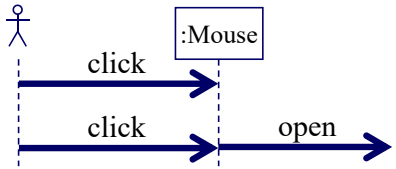


Duration Constraints

- ***Example 1:***
  Single click

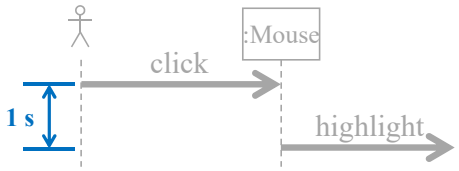


- ***Example 2:***
  Double click

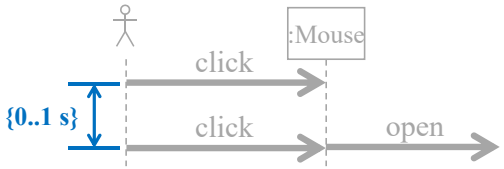


Duration Constraints

- ***Example 1:***
  Single click

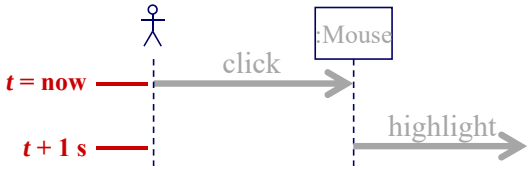


- ***Example 2:***
  Double click

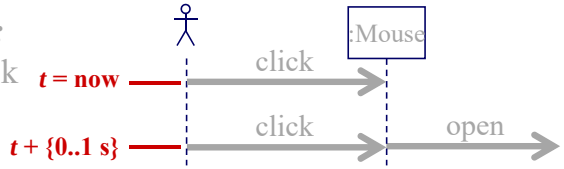


Time Constraints

- ***Example 1:***
  Single click



- ***Example 2:***
  Double click



Control Information

CombinedFragments

- Used to describe control information in sequence diagrams
- They are only for modeling ***simple*** combinations of scenarios .

Control Information

CombinedFragments

- Alternatives (*alt*)
- Option (*opt*)
- Parallel (*par*)
- Critical Region (*critical*)
- Iteration (*loop*)
- Continuation .

CombinedFragments

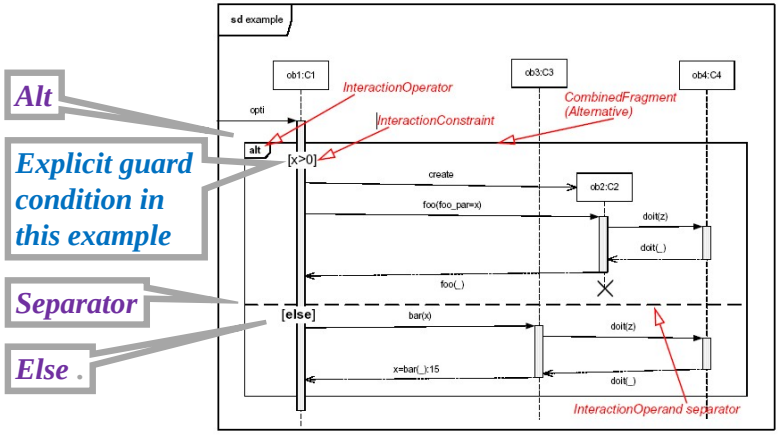
Alternatives (*alt*)

- Represents a choice of behavior
- At most one of the operands will be chosen
- The chosen operand have an ***explicit*** *or* ***implicit*** guard expression that evaluates to true

Like ***"if then else" in programming*** .

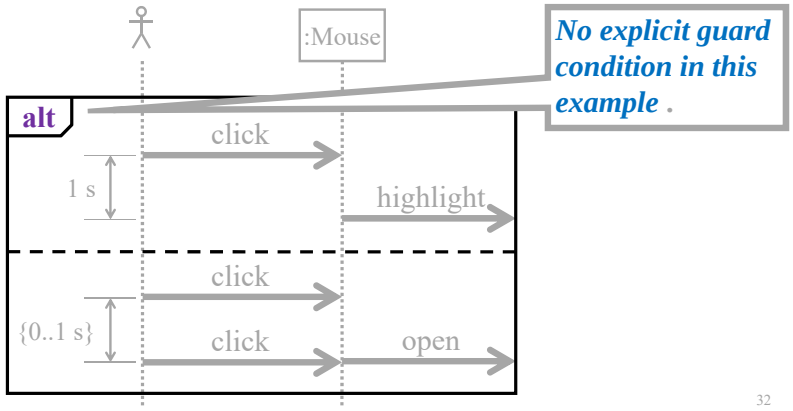
CombinedFragments

Alternatives (*alt*)



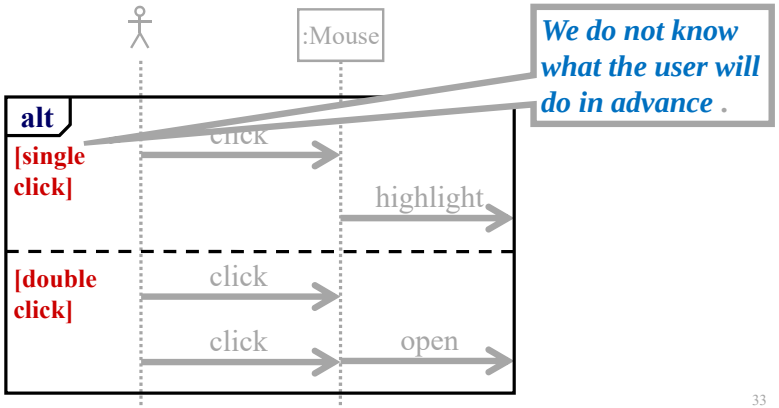
Alternatives

Single and Double Clicks



Alternatives

We Learn from Mistakes



CombinedFragments

Option (*opt*)

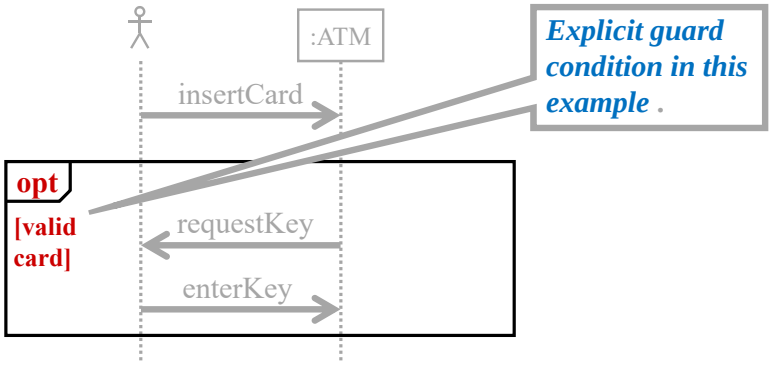
- Represents a choice of behavior where
  - either the (sole) operand happens
  - or nothing happens

Like ***"alt" where the second operand is empty***

Like ***"if then" without "else" in programming .***

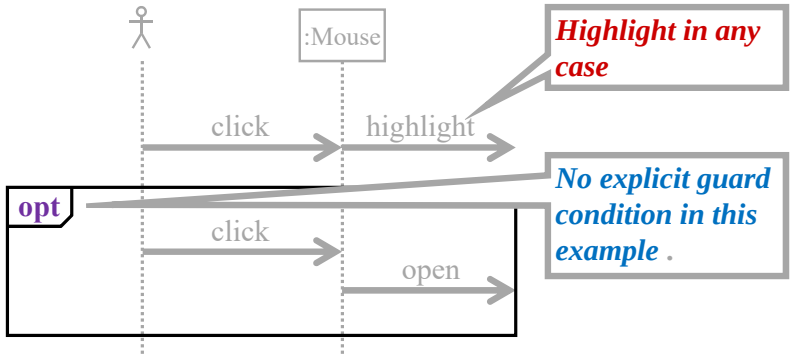
CombinedFragments

Option (*opt*)



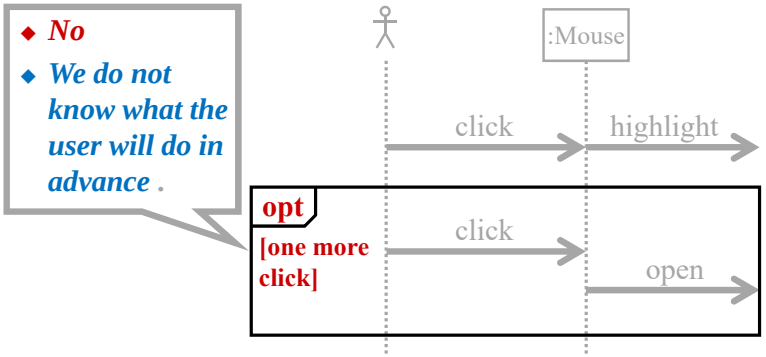
CombinedFragments

Option (*opt*)



Alternatives and Option

We Learn from Mistakes



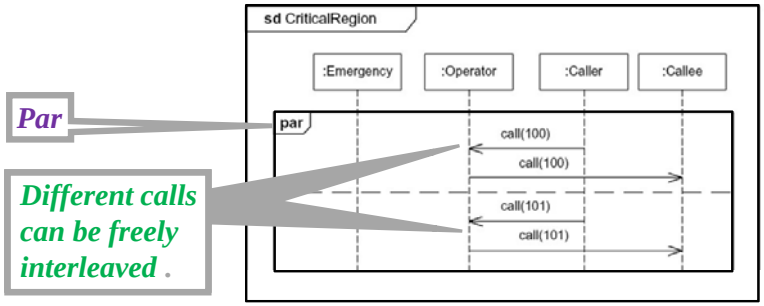
CombinedFragments

Parallel (*par*)

- Represents a parallel merge of the behavior of the operands
- Different operands can be interleaved in any way as long as the order within each operand is preserved .

CombinedFragments

Parallel (*par*)



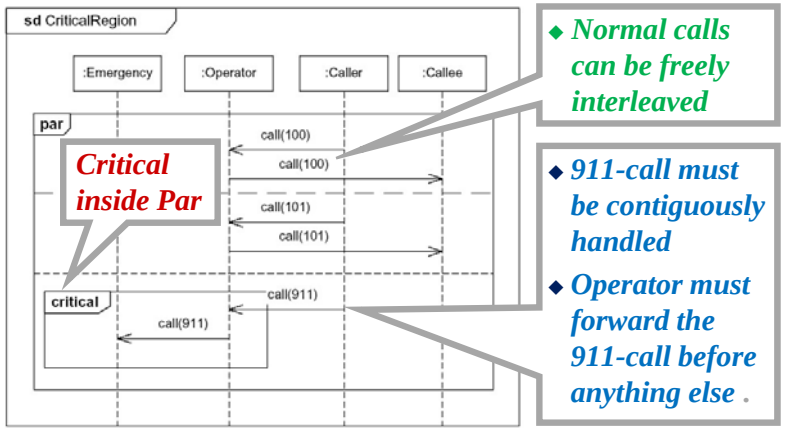
CombinedFragments

Critical Region (*critical*)

- Events in a critical region must be handled first
- Events in a critical region cannot be interleaved with events outside the region .

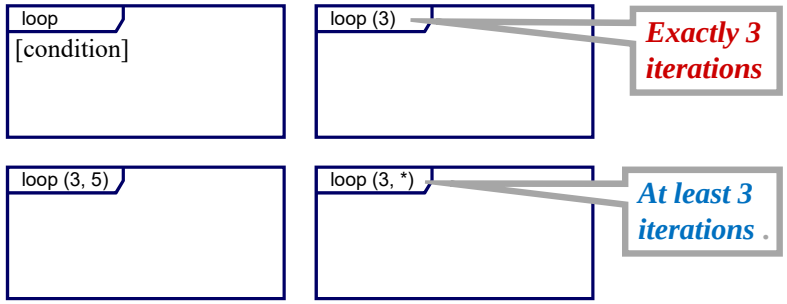
CombinedFragments

Critical Region (*critical*)



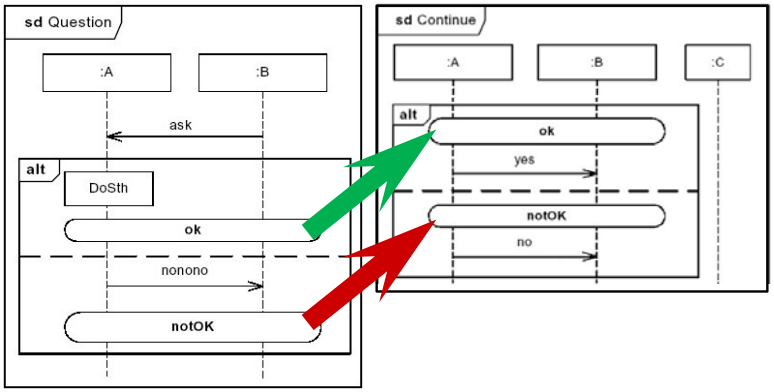
CombinedFragments

Iteration (*loop*)



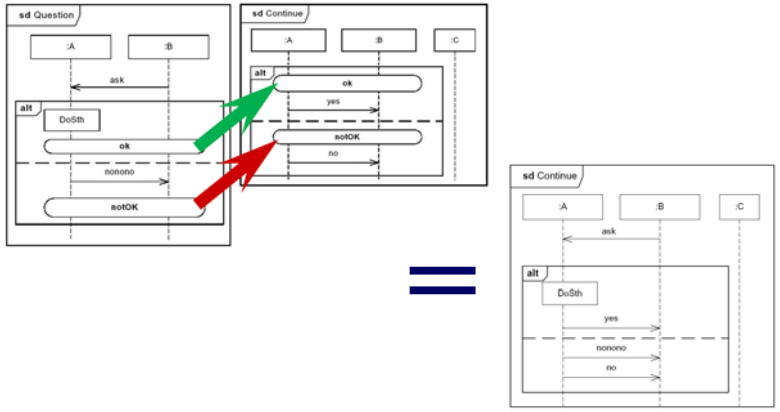
CombinedFragments

Continuation



CombinedFragments

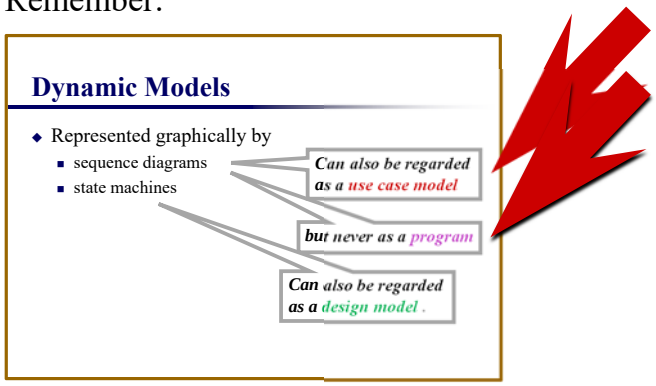
Continuation



CombinedFragments

Warning

- Remember:



CombinedFragments

Warning

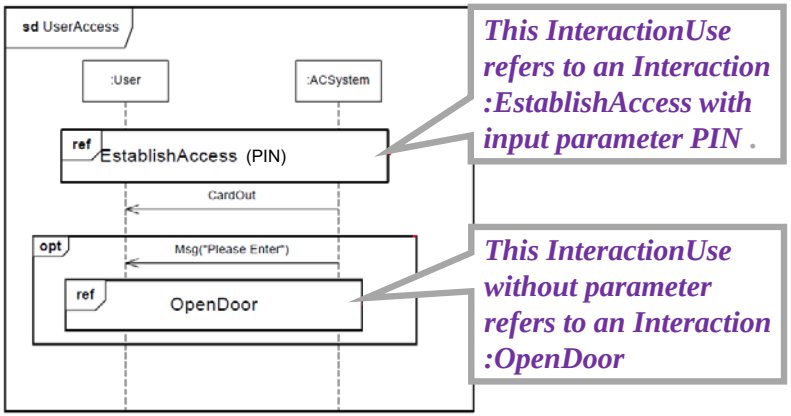
- The modelling of real-life applications are usually much ***simpler*** than my examples
- Consider drawing ***several*** diagrams for modeling complex combinations
- Do not use sequence diagrams for detailed modelling of ***program algorithms***
  - Better done using state machines, activity diagrams, or pseudo-code .

InteractionUse (*ref*)

- An InteractionUse ***refers to*** an Interaction
- The InteractionUse is a shorthand for copying the contents of the referred Interaction to where the InteractionUse is
- The copying may involve substituting parameters with actual values .

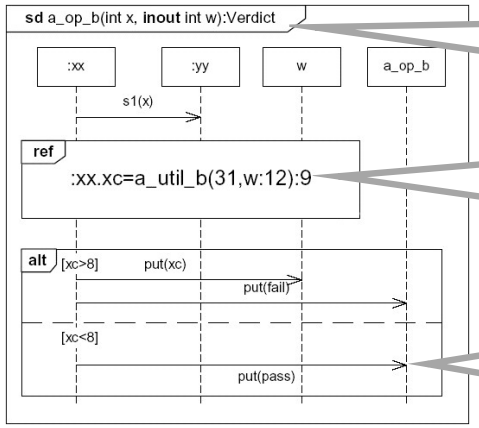
InteractionUse (ref)

Example



InteractionUse (ref)

Too Complicated Use



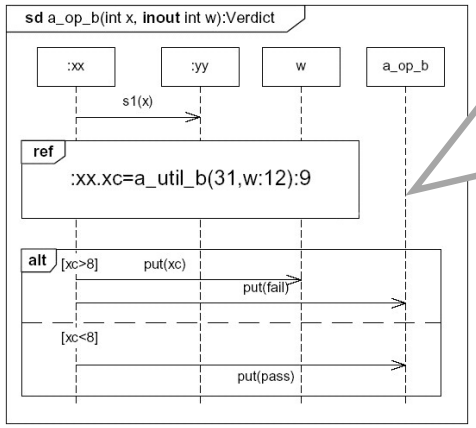
Interaction ***a_op_b*** with input integer parameter ***x***, input/output integer parameter ***w***, and return parameter ***Verdict***

- Refer to ***a_util_b*** with input **31** and input parameter w with value **12**
- Return value **9** to the attribute ***xc*** of ***:xx***

The return value from the Interaction ***a_op_b*** is passed to Lifeline ***a_op_b*** .

InteractionUse (ref)

Warning



- ***Do you find this example difficult?***
- ***Your users find it even more so***
- ***Sequence diagrams are not for Programmers***
- ***Avoid combining many scenarios together*** .